

Charlie Miller On Hacked Batteries, Cloud Security, And The iPad

Accuvant Labs' Charlie Miller talks to Tom's Hardware about the Defender's Dilemma, the security of data in the cloud, looking for vulnerabilities in notebook batteries, and the ramifications of using Apple's iPad in an enterprise environment.

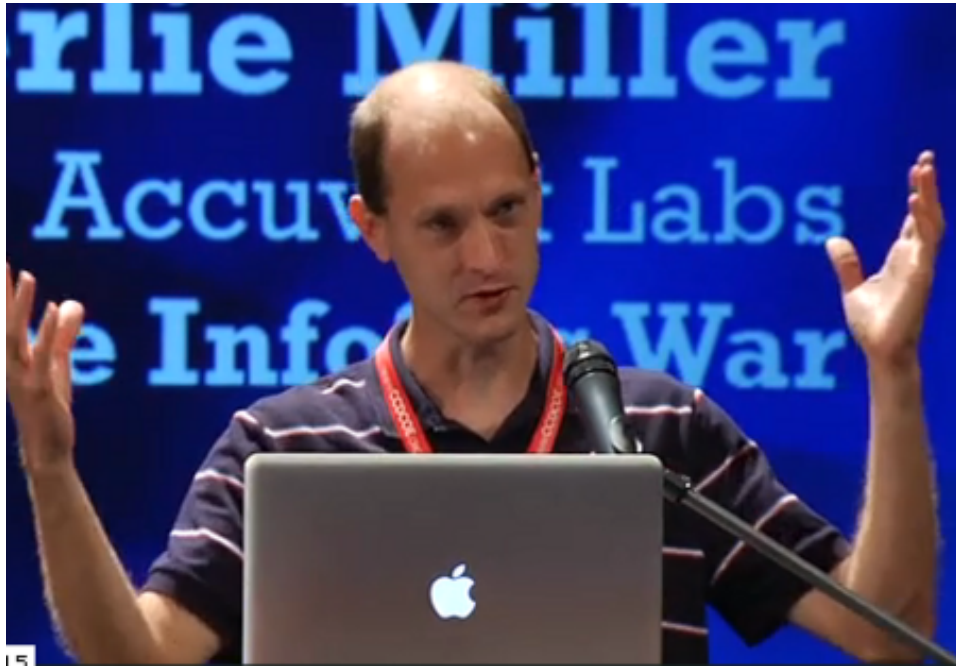
If you're not already familiar with Charlie Miller, check out [Behind Pwn2Own: Exclusive Interview With Charlie Miller](#) and [Hacking The iPhone, iPod, And iPad With A Web Page](#), two of our previous interviews with him.

Alan: Hi Charlie, thanks again for taking the time to sit down and talk with me and the readers of Tom's Hardware. I know how busy it gets around Black Hat.

Charlie: Yep, between my "day job" doing consulting and writing slides and finishing research, this is my second busiest time of the year. The only worse time is right before Pwn2Own! I can't wait until Black Hat and DEF CON are finished to relax and start some new research.

Alan: A lot has changed since the last time we chatted. The impact and critical importance of computing security has really just begun to be appreciated by mainstream users. The New York Times had a great feature on the development of Stuxnet and detailed how the good guys undermined and crippled Iran's nuclear program. We've seen the bad guys attack Lockheed Martin through a targeted effort that began with compromising RSA SecurID. Lastly, we've seen end-users directly impacted by the actions of groups like Anonymous and LulzSec. It's the good, the bad, and the ugly.

So for today, I'm hoping I can get your thoughts on some big-picture stuff before I pick your brain on the new iOS jailbreak and the battery firmware vulnerability.



Charlie: Well, who the bad guys and who the good guys are can be a bit hard to determine sometimes. I'm sure the Iranians don't consider Stuxnet to be a force for good. But yes, let's talk about the big picture.

Alan: Well, I know I'm a good guy, and you're a good guy. People can trust us. We're doctors. Anyway, after the Brighton Bombing in '84, the IRA released a statement that included the line "...remember we only have to be lucky once. You have to be lucky always." When it comes to computing security, it seems like it's the same challenge. Only in this case, it's even worse. The bad guys are coming from multiple fronts. You have targeted attacks, automated botnets, and broad social engineering spam. You also have different motives ranging from espionage and financial or political gain to activists looking to make a statement. While there was a political process that could bring peace to the United Kingdom, you're not going to be able to negotiate with someone looking to steal credit card info or sensitive data.

Can we actually win this war or are we just hoping to minimize our losses?

Charlie: Yes, we call it the Defender's Dilemma. Defense is always harder because you have to be perfect, where attackers only have to find one flaw. This is why it's so much more fun to attack Apple than to work for Apple!

I have to say, things are a bit bleak when you put it that way. There will always be vulnerabilities and there will always be criminals, so it's hard to figure the way out. Especially as end users there is almost nothing you can do; you have to rely on the security of the software you run and have little control over how secure it is. As a society, we cannot eliminate computer attacks. However, what we can do (and this is the approach the industry is sort of taking) is make it so hard and expensive to pull off attacks that it becomes economically infeasible for most attackers. And even for those with the expertise to still pull off the attack, it minimizes the number of attacks they can perform. The way we make it more difficult is to reduce the number of vulnerabilities and ensure users' software is up to date and "secure by default". Also, make the OS resilient to attack with things like stack canaries, ASLR, DEP, and sandbox applications so that multiple exploits are needed. We also need to better control the software loaded on our devices (i.e. Apple's App Store model). So, instead of having to write a single exploit, it takes three or four in order to perform an attack. This means most attackers won't be able to pull it off, and those who can will have to spend much more time working it out.

Alan: What about "mathematically proving" that software is "correct" and performs as expected to meet the design requirements? That's done with avionics software. Can we do that with regular software? Can you mathematically prove that something is secure, or at least impervious to specific attack patterns like fuzzing or SQL injection?

Charlie: It's probably possible sometimes, but it is not done. We're still really in the stone ages of software security. At this point, the only practical thing to do is fuzz, audit, and analyze the hell out of it. Microsoft fuzzes everything, but obviously there are still plenty of bugs in its stuff. I've found critical bugs in software that has been analyzed by static analysis tools. Research indicates that different fuzzers find different bugs. Finding all (or even most) critical software vulnerabilities is really hard, time intensive, and expensive. NASA might have the time and money to make sure the software on their Mars Rover is perfect, but software vendors want to ship software and make money and are willing to live with a "few" vulnerabilities.



Alan: Besides better software, what about hardware issues? Joanna Rutkowska published the SMM attack a couple of years ago, and you recently talked about the firmware attack with Apple batteries. How do we approach this problem?



Charlie: This is really hard. Another example you left out is Ralf Phillip Weinmann and his mobile baseband attacks. There are lots of different chips in all of our electronics that you don't think about. This is one of the reasons I was interested in the battery research. The worst thing about hardware is that it is hard/expensive to analyze. We can all download Internet Explorer and audit the code/fuzz it. But it takes equipment and special skills to look at hardware. I probably spent \$1000 for equipment on the battery research and that was just for fun. These barriers make the systems less secure because it discourages researchers like me from analyzing it.

Alan: Where does cloud computing fit into this? You're putting a lot of trust into the company developing the cloud software and the company actually hosting the cloud. If its software is bad, or worse, if its privacy policies are incomplete or its employees are unethical, you are at a significant level of risk for data compromise. In addition, a big database like that would be a prime target for hackers. On the other hand, companies like Amazon, Apple, Microsoft, and Google should be better-equipped than the average end-user when it comes to security.

Charlie: Yes, cloud security is tough because it can't be independently validated very easily. We can all tear apart MS PowerPoint to see what it does with our data, but when you ship your data off to the cloud, researchers like me cannot look at the software to try to find bugs. In fact, poking around on their Web site is illegal. Using software that is not on your system, and thus cannot be torn apart and reverse engineered, means you are putting a large amount of trust on whoever is writing that software. The guys like me won't be able to help you. As for whether these big companies are better than the average person, I'm not sure. Sony might be a good counterexample to your argument.

Alan: What about a Consumer Reports-type third-party to grade companies on their security? Even if I could do a better job than Sony, the guys at Microsoft and Google definitely are better than the average user. The average user isn't going to be able to do what Google did with counter-hacking the Chinese hackers or what Microsoft did

with Waledac which combined technical measures with legal/political measures.

Charlie: Yes, this is one of the solutions I recommended during my recent talk at the NATO Cooperative Cyber Defense Centre of Excellence in Estonia. On a high level, something like Underwriters Laboratories. If you buy a toaster, and it has the UL seal on it, you can be sure it won't burn your house down. We need something like that for software where if you see the UL seal, you know it might not be perfect, but it has undergone and passed a certain level of scrutiny. On a technical level, I could imagine something like a private fuzzing test suite that a product would either pass or fail, and the software maker would not be given the failing test cases. In this way they couldn't "train for the test." They'd probably find way more bugs than the private test suite would find, just in the effort to pass the test.



 Zoom

Alan: Say you have secure hardware and secure software. How do we protect against social engineering?

Charlie: People are usually the weakest link in security. Almost all of the exploits I write at least require the user to go to a malicious Web site. That means clicking on a link sent via email, surfing on a public wireless access point, etc. Computers are designed, for the most part, to do things we ask them to. No matter how much security you build into a system, if the user really wants to run a piece of malware they think will show them

some naked pictures, they're going to figure out a way to run that program.

Alan: Let's talk a little bit about the iPad jailbreak. From what I understand, this is another PDF-based exploit. Have you had a chance to look at this?

Charlie: Yes, I've reverse engineered it a bit. The exploit is delivered via a PDF file, but the underlying vulnerability is in how it parses a font that is embedded in the PDF. This "malicious" font could have been delivered in ways besides PDF files. Anyway, it is a very clever exploit. The bug is in this little state machine that is processing the font. The bug allows the attacker to change where the program thinks the end of the buffer where the state machine is operating is located to beyond where it is supposed to be. Then the state machine can operate on parts of memory it is not supposed to while processing the font. This allows it to corrupt memory (to get control of the process) as well as read and operate on values from memory (which allows it to bypass ASLR, allowing it to find some executable code to use). At that point, it reuses the existing code fragments it wants (this technique is called return oriented programming) to launch a second exploit against a different vulnerability to escape the iOS sandbox, get root, disable code signing, and finally jailbreak the phone.

Alan: How could Apple have prevented this?

Charlie: Have fewer bugs I suppose! This particular bug would have required a code audit, I think. Fuzzing probably would not have found it. Also, it could have reduced the attack surface available for the second, escape-the-sandbox vulnerability. Everything else, it did right. ASLR, DEP, sandboxing. The iOS security model is pretty good, but this just goes to show that there are always potentially attacks that can work.

Alan: The mainstream media often talks about "jailbreaking," but the term really downplays the underlying issue that this is a remote exploit that allows arbitrary code to be run. As I see more and more enterprises using iPads, I can't help but be paranoid about the security issues. We know that over 114 000 email addresses of early iPad owners were stolen from AT&T. These early adopters included high-level executives at major technology and finance/banking firms as well as government agencies. Given that it's trivial to remotely execute code via PDF engine, and the documented sophistication of the hacks of companies like Lockheed Martin, it seems almost naïve to think that no one has attempted to compromise sensitive data via targeted attacks on

the iPad. How should we deal with this issue?

Charlie: Well the problem is that all devices are susceptible to attacks of this nature, and an iPhone/iPad is a device. iOS-based devices are more secure than Mac OS X-based devices due to the code signing and sandboxing of applications. They are probably more secure than desktops running Windows 7. The biggest risk is you'll physically lose them and lose your data that way. But, despite the fact we've seen attacks against iOS devices, such as jailbreakme.com, it is pretty rare and malware is very rare too. I think iOS is about as good as we can do for now. There is always risk your device will be compromised. What you need to focus on is limiting access to data at any one time, detecting attacks quickly, etc.

Alan: Along the same lines as the earlier cloud computing question, are organizations better off adopting a heterogeneous computing environment or a homogeneous one? That is, if I only have one platform to support (say, an iPad), a security officer can really focus all of the efforts on securing one platform. If I have to support multiple platforms, my efforts to secure the network will be diluted across systems and the sieve will be twice as big with more potential holes and vulnerabilities. On the other hand, the argument for heterogeneous computing is that if I do lose against the bad guys and one of my platforms is compromised, I can quickly switch/rollover the company to the still uncompromised platform (and make the bad guys work twice as hard).

Charlie: This is a great question and the same answer might not fit for everyone. I used to recommend homogeneous environments to ease burden on patching systems. If your enterprise can't really keep up on patches, this is probably for you. However, for the best defense, heterogeneous networks are superior. You must design your network knowing that machines on it will be compromised. Desktops will get malware, your CMS will get SQL injected, etc. Just like you shouldn't use the same passwords in multiple places, you shouldn't use the same operating systems or devices in different places. That way, it will be much harder if an attacker needs exploits against different platforms to make any progress. In fact, most attackers won't have the skills to attack two or three different up-to-date systems, which raises the overall bar for security. You don't want every single computer in your network exploited because of a single Windows kernel remote.

Alan: Let's talk about the battery exploit. How did you even come up with the idea

about looking for vulnerabilities in the battery?

Charlie: At Black Hat last year I saw Barnaby Jack's ATM hacking talk and thought the coolest thing about it was how you could explain what he did to someone with no technical know-how. "You see that ATM? I can make it spit out money." I wanted to work on something like that and thought about the risks of battery safety for laptops. I set out to see if a remote attacker could blow up my laptop. I still don't really know the answer to that question, but I do know that 1) attackers can certainly get far into that subsystem and 2) I can't blow up a battery :) It was a fun (but long) project because I don't know that much about hardware, so I had to learn a lot as I went.

Alan: This exploit would be resistant to reformatting, right? The ultimate pre-boot malware.

Charlie: So, one of the things I show you can do is make modifications to the firmware that runs on the main chip on the smart battery. You can make it do whatever you want because Apple used default passwords on the chips (made by Texas Instruments). Code you put there would survive reinstallation of the OS, new hard drives, new motherboards, and so on. However, the code cannot directly affect the OS or hard drive, so in order for it to be malware, it would have to attack the OS through some kind of vulnerability in the way the OS handles messages from the battery. Now, I don't know if such a vulnerability exists, but I do know that whoever wrote that code wasn't thinking that the battery would be sending malicious data, so I wouldn't be surprised to find one!

Alan: What about systems implementing trusted boot and things like Intel Trusted Execution Technology? Could that have prevented this attack?

Charlie: No, that wouldn't help. The boot process would all be fine and dandy and after the OS was up and running, the battery would attack it (if such a vulnerability exists) and then inject code.

Alan: When you or any other security researcher discovers system vulnerabilities like this, it's natural for people to assume that this is the "first discovery" of the problem. Indeed, often times it is only days after a vulnerability is reported that attacks show up based upon the newly-published vulnerability. But we know the bad guys are talented. The bad guys may actually have more money behind them. As the stakes get

higher, when do we begin to assume that the bad guys have beat us to discovery and that any vulnerability that is reported is already actively being exploited and we just didn't know?

Charlie: This is a really interesting question. I'm always worried that other researchers are going to discover the same things I discover before me. In fact, I had a Mac OS X exploit ready to go at Pwn2Own this year and didn't get a chance to use it because someone else beat me to it. Then, a few days later, Apple patched it, so someone else had independently found it (or pwned me and stole!) That was something I liked about the battery research. because I thought nobody would ever think of this wacky idea and I could take my time looking at it. But it turns out that Barnaby Jack (the ATM hacking guy I mentioned earlier) had looked at exactly the same thing and discovered many of the same things I found about a year ago and never told anyone because he didn't catch his laptop on fire. So no matter how clever you are, the odds are that somebody else already knows how to do what you're trying to do. People think I find good stuff, but I'm one guy doing this in the evening for fun with no budget. Compare that to all the money the U.S. government (or China) spends on cyber security. It is hard to imagine they don't know some things we haven't figured out yet.

Alan: Usually, at the end of the interview, I ask security experts what end users should do to be as secure as possible. I know that one recommendation I'm making is to abandon the traditional thinking of "don't update your software right away, so that other people can be the beta testers and figure out the compatibility problems." It seems more prudent to always update to the latest version to keep yourself patched against the newest vulnerabilities and deal with the compatibility issues as they come. But it seems like in today's world, the end-user is playing a less important role. The end-user with the latest software updates who is also savvy to social engineering cannot protect himself against hackers who steal credit card data from Sony. From a criminal organization, it's far more effective to try to attack large databases rather than individual systems. What should be the call to action in 2011?

Charlie: Yes, as individuals we are pretty powerless. Even enterprises have to rely on the security of their devices and desktops, which they have little control over. Enterprises buy IDS, AV, etc., but still can get attacked by zero-days and it's all over. So what we really need to do is force large vendors to do a better job writing secure software. Either

lobby our government to hold them responsible when their bugs cause us financial loss or vote with our pocketbooks. Refuse to buy software that has problems, require the software to be audited and fuzzed by some independent organization. Besides that, all we can do is wait for the inevitable and then try to react as quickly as possible to limit the damage.

Alan: As always, I really enjoyed talking with you and appreciate your insights.

Charlie: Thanks. I enjoyed it as always!

We thought Charlie's recent keynote at NATO's International Conference on Cyber Conflict was pretty interesting. Check out the full discussion below.

Charlie Miller's Keynote At NATO's ICC